07

Mejorando el controlador

Transcripción

[00:00] Hola a todos y todas, bienvenidos y bienvenidas a este nuevo video sobre servlet. En este caso queríamos agregar algunas mejorías de código sobre nuestras implementaciones. No vamos a agregar alguna cosa que sea específica de servlet, sino que vamos a implementar buenas prácticas a nivel de en el momento de codar.

[00:23] Entonces miren lo siguiente, quería mostrarles esto, ¿nosotros qué es lo que realizamos dentro de cada uno de estos if else? Dentro de cada if else nosotros tenemos nuestra instancia de una clase y luego llamamos al método ejecutar. Instancia de una clase, llamamos el método ejecutar, instancia, ejecutar, todo siempre lo mismo.

[00:44] Lo hemos hecho de esta forma, para poder aplicar un patrón que ahora enseguida vamos a ver. Y no solo eso, sino que nosotros también hemos hecho que nuestros strings, espero que hayan seguido de la misma forma en que yo estuve realizándolo, en nuestros strings de cada una de las acciones que nosotros colocamos en la URL, son el mismo nombre que tenemos en nuestras clases.

[01:08] Entonces tenemos mostrarEmpresas, que es el mismo nombre que la clase mostrarEmpresa, entonces así es con todo. Entonces con esto, ¿cómo es que yo voy a querer solucionar este problema de los if else eternos que si yo quiero agregar una nueva acción, voy a tener que agregar un nuevo else if acá?

[01:29] Bueno, ¿qué es lo que me gustaría hacer? A mí me gustaría hacer una cosa por el estilo, ya que nuestro nombre del string es igual el nombre de nuestra clase, entonces me gustaría hacer algo por el estilo, algo así como paramAcción.ejecutar. ¿Por qué? Porque paramAccion, ese string, es el mismo nombre que mi clase y yo después realizo la instancia de la clase .ejecutar.

[01:56] Entonces, ¿cómo podemos hacer esto? Bueno, si yo quiero llamar a ese método ejecutar, ese método, ese método pertenece a un objeto, a una instancia de una clase y una instancia de una clase necesita sí o sí una clase para ser instanciada. Entonces, ¿cómo nosotros, reconocemos a una clase?

[02:21] Nosotros reconocemos una clase a través de su nombre listaEmpresas. Cada una de estas acciones es una clase con un nombre, entonces lo primero que podemos hacer es, por ejemplo, colocar lo siguiente, obtener el nombre de la clase que nosotros vamos a querer. En este caso vamos a hacer un string nombreDeClase = ¿a qué es igual? A lo que nosotros recibimos como parámetro de acción.

[02:44] Entonces va a ser la misma cosa, entonces vamos a colocar paramAccion; ¿Pero qué es lo que ocurre? Una clase no se compone solamente de su nombre. También se compone de las carpetas dentro de la cual está esa clase, listaEmpresas no es solamente listaEmpresas. Es com.alura.gerenciador.accion.ListaEmpresas.

[03:08] Entonces vamos a agregar eso, ya que todas están dentro del mismo paquete, vamos a agregar eso. Vamos a colocar "com.alura.gerenciador.accion." y luego de eso, agregamos el nombre de la clase. Con esto, nosotros ya tenemos una referencia al nombre de la clase. Ahora eso es solo un string. ¿Cómo hacemos para obtener la clase en sí misma?

[03:41] Bueno, lo que nosotros podemos utilizar es un una clase llamada class. Muy extraño, pero funciona. Class es un una clase genérica, y toda clase tiene

una herencia de ese class, entonces básicamente toda clase que nosotros creamos es también al mismo tiempo una class.

[04:05] Entonces nosotros vamos a obtener una clase. Entonces class clase = ¿Cómo hacemos para obtener exactamente? Por ejemplo, cuando llega acá listaEmpresas, nosotros tenemos com.alura.gerenciador.accion.ListaEmpresas. ¿Cómo hacemos para obtener una referencia de esa clase? Nosotros vamos a utilizar el siguiente método, o sea, class.

[04:29] Vamos a usar un método que está dentro de nuestra clase class.forName. Con esto nosotros, eso está bien, forName y enviamos nuestro nombre de la clase, con ese método nosotros obtenemos una referencia a la clase que queremos, ven que es bastante genérico. En este caso podemos obtener listaEmpresas, pero podemos obtener mostrarEmpresas. Para cualquiera de esos funciona.

[04:56] Entonces vamos a tener una referencia a esa clase en memoria. Va a quedar ahí en memoria. Cada vez que nosotros hacemos, por ejemplo, un listaEmpresas new ListaEmpresas, por detrás, la máquina virtual de Java obtiene una referencia a la clase y crea una instancia.

[05:14] Entonces lo que vamos a hacer es, primero tenemos esa referencia a la clase y ahora lo que queremos hacer es crear una instancia de esa clase, ese new que nosotros hacemos acá, ¿entonces cómo hacemos eso? Cuando nosotros creamos una instancia, nos devuelve un objeto, entonces nosotros vamos a tener un object obj = clase, vamos a referenciar a la clase, punto. Y acá vean que tenemos muchos métodos.

[05:48] Todos estos métodos son dados a través de una API de la propia clase y todo para poder traer más información sobre todos estos métodos, ustedes tienen que buscar, por ejemplo, el tema de reflexión. Esto se llama reflexión, e

poder tener todas estas referencias, estos métodos disponibles genéricos de clases genéricas.

[06:13] Entonces en este caso dentro de todos estos métodos vean que nosotros podemos obtener el constructor, podemos obtener cuáles son los campos, las variables que tienen dentro de esa clase, interfaces, muchas cosas, lo que nosotros vamos a querer es este newInstance. Ese newInstance es lo que lo que nos permite obtener una instancia de ese objeto, es lo mismo que ese new, ese new de acá.

[06:48] Bueno, vean, esto está tachado. ¿Por qué? Significa que está diprecated o depreciado. Esto puede significar que en el futuro esto ya no sea posible de realizar o que en el futuro, tal vez en versiones futuras de Java no puedan hacer ese newInstance, pero por ahora nosotros podemos, y no es el foco de esta clase el poder corregir eso que es un tema más profundo que el que estamos viendo.

[07:25] Por ahora, su instante para nosotros nos está sirviendo para lo que queremos, obtener una nueva instancia. Perfecto, ¿ahora yo qué es lo que quiero? Nosotros ya obtuvimos ese new de nuestra ListaEmpresa, en el caso de ListaEmpresa, y después de eso, vean que nosotros realizamos un ejecutar y lo colocamos dentro de una variable string. Entonces nosotros vamos a hacer eso

[07:51] Vamos a hacer un string nombre = obj.ejecutar, y tengo que pasar como parámetro los request y response.¿Qué es lo que ocurre? Nos está dando acá un alerta en rojo. ¿Por qué? Porque nos está diciendo que obj es un objeto genérico, este ejecutar no existe dentro de cualquier objeto dentro de cualquier clase. Ese ejecutar es propio de cada una de nuestras clases que nosotros hemos creado.

[08:34] Ese ejecutar solo existe acá en nuevaEmpresa, en ListarEmpresas, etcétera, bien entonces es nuestro. ¿Cómo hacemos para poder ejecutar ese

método ejecutar dentro de un objeto genérico? Bueno, lo que nosotros tenemos acá que nosotros hemos realizado es un contrato.

[09:00] Aunque no lo parezca, cada una de estas, de listaEmpresas, nuestro mostrarEmpresa, eliminarEmpresas, todos ellos tienen un método llamado ejecutar, entonces es como que están cumpliendo con un contrato implícito que nosotros no lo hemos formalizado. Todos ellos tienen ejecutar.

[09:19] Entonces todos cumplen un contrato. Vamos a formalizar ese contrato. ¿Cómo hacemos para formalizar un contrato? En este caso nosotros lo que utilizamos es una interfaz, una interfaz que va a ser implementada por cada una de esas clases. Bien, entonces vamos a hacer eso, vamos a ir a nuestro paquete de acción, clic derecho, newInterface. Y vamos a llamarlo de acción. Acá vamos a tener nuestro contrato.

[09:51] Bien, ahora ese contrato, ¿qué es lo que va a tener? Va a tener un método, ese método es el método que hemos estado usando en todas nuestras clases que es el método ejecutar. Entonces vamos a copiar esta línea public String ejecutar, todo eso, copiamos bien, vean que ya ha hecho el importación, algun import, no vamos a necesitar ese public porque ya por definición, todo método que colocamos dentro de acción va a ser accedido por cualquiera que lo implemente. Punto y coma.

[10:28] Y vean que no tenemos un cuerpo, que no tenemos. ¿Cómo se dice? Una implementación de ese método. Bien. "Ctrl + S" y ahora lo único que vamos a hacer es que cada una de estas clases implementen esa acción. Implements Accion, bien perfecto. "Ctrl + C". Y vean que no hubo ningún error. ¿Por qué? Porque ya todos esos ya estaban ejecutando ese ejecutar.

[11:00] Entonces no hubo ningún problema de código. "Ctrl + S" y vamos a hacer lo mismo para cada una de nuestras acciones. Implements, "Ctrl + S", modificarEmpresa, "Ctrl + S". Con esto estamos formalizando el contrato que

nosotros ya teníamos del método ejecutar. "Ctrl + V", "Ctrl + S". Voy a cerrar cada uno de estos, form acción, eliminar, modificar, mostrar y nuevaEmpresa.

[11:36] ¿Por qué hice eso? ¿Por qué formalicé ese contrato? Porque ahora, como todos todas esas clases tienen, yo puedo asegurar que todas ellas tienen el método ejecutar gracias a este contrato llamado acción, que es interfaz, entonces yo puedo decir, por ejemplo, que este objeto es también una acción, entonces puedo hacer acción acción, también iguala, voy a hacer un casting.

[12:08] Acción obj. Entonces estoy convirtiendo "Ctrl + Shift + O". Estoy convirtiendo ese objeto que es genérico a uno que es específico, que ya sabemos que él ese objeto específico que está implementando acción, tiene el método ejecutar. Acá está dando un error. Es porque estoy llamando a obj y no a acción.

[12:37] Si yo acá cambio a acción, ahora si tenemos el método ejecutar. Bien, entonces con esto nosotros hemos podido hacer que cada vez que nosotros llegamos a una URL que representa a una clase dentro de nuestro código, nosotros vamos a poder instanciarla y ejecutar la acción. Déjenme que voy a dejar todo esto un poco más limpio, este código más limpio.

[13:10] Yo podría unir estas dos líneas directamente a hacer el casting a nuestro obj acá. Esto lo elimino, "Ctrl + S". Y veamos, nos está diciendo que es un tipo genérico, todo bien. Y ahora una cosa que tenemos que hacer es colocar un tryCatch acá, ya que este casting en particular y dentro de ese forName, en caso de que no pueda encontrar esa clase y no pueda hacer el casting, nosotros tengamos un respaldo y no quiebre todo nuestro servidor.

[13:54] Entonces, para realizar eso, nosotros vamos a seleccionar todo este código, estas tres líneas de código, vamos a hacer un clic derecho surround with, try/multi-catch block. Acá, por ejemplo, nos va caso no encuentre la clase, caso tenga una excepción al intentar realizar la instancia de esa clase, etcétera, nosotros vamos a estar protegidos.

[14:21] Entonces cuando ocurra eso yo no quiero hacer una implementación específica. Caso ocurra cada una de esas cosas, simplemente voy a delegar la tarea de mostrar la excepción a nuestro ServletException. Yo no puedo poner esas excepciones acá porque este service ya está cumpliendo un contrato en el cual solamente puede tener esos throws, tirar esas excepciones.

[14:46] Pero en este caso yo lo que puedo hacer es darle a nuestro servletExceptcion nuestro mensaje. Entonces vamos a hacer un throw new ServletException y vamos a enviarle nuestro mensaje (e), "Ctrl + S". Vamos a probar. Antes de probar vamos a comentar todo este código, vamos a comentar. Voy a comentar todo esto, esa lista de métodos, de else if y esto de esta parte de string y tipo de direction lo voy a llevar para acá arriba.

[15:31] Ese string nombre no lo necesitamos. Voy a llevar este string, lo que sí este string nombre lo voy a llevar para acá, arriba, punto y coma, ahí está. Bien, voy a dar algunos espacios solo para mejorar un poco acá, acá es nombre. "Ctrl + S". Muy bien. Vamos a ver si esto está funcionando, voy a reiniciar el servidor.

[16:08] Vamos a ver acá a ver si funciona entrada?accion=ListaEmpresas, nuestro listaEmpresas está funcionando, por lo menos. Vamos a ir a modificar Aluraaa, enviar, eliminar. Esto está funcionando y nuestro último, la última entrada que tenemos que probar es NuevaEmpresaForm. Vamos a ver si está todo bien. Enviar, perfecto. Está funcionando todo.

[16:33] Entonces con esto vemos que nuestra interfaz está funcionando. Gracias a nuestra interfaz hemos podido crear un contrato que todas nuestras acciones están cumpliendo. Todas tienen el método de ejecutar, entonces cómo hemos hecho eso, nosotros podemos obtener el nombre de la clase directamente desde nuestra acción, desde nuestra URL, dentro de los parámetros que recibimos.

[17:00] Creamos una referencia a la clase en memoria, instanciamos esa clase y hacemos el parsing y luego ejecutamos el método. En todas ellas va a ser lo mismo. Y esta parte de acá es lo mismo que ya teníamos. Entonces con esto bueno, hemos casi terminado esta aula. Voy a hacer un resumen en el próximo video sobre algunas buenas prácticas que hemos utilizado.

[17:32] Y cualquier cosa si tienen más dudas sobre estos temas, recuerden que esto se llama reflexión. Hemos utilizado un patrón acá que no les había dicho, el patrón comando, que es encapsular nuestra clase y perdón, encapsular nuestro método ejecutar, colocarlo dentro de una clase y simplemente realizar